Parallelization of iterative and direct schemes for keller-box method on distributed memory platform

Norma Alias
Ibnu Sina Institute
Universiti Teknologi Malaysia, Skudai
Johor, Malaysia
norma@ibnusina.utm.my

Norhafiza Hamzah, Norsarahaida S.Amin, Roziha Darwis
Department of Mathematics, Faculty of Science
Universiti Teknologi Malaysia, Skudai
Johor, Malaysia
norhafizahamzah@gmail.com

Noriza Satam, Zarith Safiza Abdul Ghaffar
Department of Mathematics, Faculty of Science
Universiti Teknologi Malaysia, Skudai
Johor, Malaysia
norhafizahamzah@gmail.com

***Abstract*—In this paper, we present iterative schemes, specifically the conjugate gradient, and Gauss seidel red-black (GSRB) and direct schemes namely LU factorization and Gauss elimination for Keller-box scheme. The aim of this paper is to offer reasonable assessments and contrasts on behalf of the numerical experiments of these two schemes ported to run through Parallel Virtual Machine (PVM) on distributed memory platform. The computational complexity also presented for the comparison purpose, and the graphs of parallel evaluation in terms of speedup, efficiency, effectiveness and temporal performance are presented as well.**

Keywords-iterative; direct; parallel; keller-box;

I. INTRODUCTION

The numerical solution methods for linear systems of equations, are broadly classified into two categories, direct methods, and iterative methods [1]. The most reliable and simplest solvers are based on direct methods, but the robustness of direct solvers comes at the expense of large storage requirements and execution times, while the iterative techniques exhibit problem-specific performance and lack the generality, predictability and reliability of direct solvers. Yet, these disadvantages are outweighed by their low computer memory requirements and their substantial speed especially in the solution of very large matrices [2]. However, direct methods have been recommended for solving large sparse linear systems when, among other reasons, the system is ill-conditioned [3]. Direct methods obtain the exact solution in finitely many operations and are often preferred to iterative methods in real applications because of their robustness and predictable behavior. Although iterative methods for solving linear systems find their origins in the early nineteenth century especially by Gauss, the field has seen an explosion activity stimulated by demand due to extraordinary technological advances in engineering and sciences [4].

According to [5], the term 'iterative methods' refers to a wide range of techniques that use successive approximations to obtain more accurate solutions to a linear system at each step. Beginning with a given approximate solution, these methods modify the components of the approximation, until convergence is achieved. They do not guarantee a solution for all systems of equations. However, when they do yield a solution, they are usually less expensive than direct methods [1].

II. PROBLEM STATEMENT

Boundary layer is a thin layer of fluid in which viscosity effects are significant, and formed along solid boundaries [6]. In aerodynamics, the details of the flow within the boundary layer are important for many problems including the skin friction drag on an object, the heat transfer in high speed flight, and wing stall which is the condition when aircraft wings will suddenly lose lift at high angles to the flow. The simplified Navier-Stokes equations, known as Prandtl's boundary layer equations are

$$\frac{\partial u}{\partial x}+\frac{\partial \upsilon}{\partial y}=0 \tag{1}$$

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+\upsilon\frac{\partial u}{\partial y}=-\frac{1}{\ell}\frac{\partial p}{\partial x}\nu\frac{\partial^2 u}{\partial y^2} \tag{2}$$

with the boundary conditions

$$y=0: u=\upsilon=0;\ y=\infty: u=U(x,t), \tag{3}$$

where the potential flow $U(x,t)$ is to be considered known [7]. A suitable boundary layer flow must be prescribed over the whole x, y region under consideration for the instant $t=0$. In the case of steady flow, the system of equations is written as

$$\frac{\partial u}{\partial x}+\frac{\partial \upsilon}{\partial y}=0 \tag{4}$$

$$u\frac{\partial u}{\partial x}+\upsilon\frac{\partial u}{\partial y}=-\frac{1}{\ell}\frac{\partial p}{\partial x}\nu\frac{\partial^2 u}{\partial y^2} \tag{5}$$

and the boundary conditions

$$y=0: u=\upsilon=0; y=\infty: u=U(x,t), \tag{6}$$

Fig. 1 shows a boundary layer along a flat plate at zero incidence. Let the leading edge of the plate be at $x=0$, the plate being parallel to the x -axis and infinitely long downstream on Fig. 1. We shall consider steady flow with a free-stream velocity, U_∞ , which is parallel to the x -axis. The velocity of potential flow is constant in this case, and therefore, $dp/dx\equiv 0$ [7]. The boundary layer (4) to (6) become

$$\frac{\partial u}{\partial x}+\frac{\partial \upsilon}{\partial y}=0 \tag{7}$$

$$u\frac{\partial u}{\partial x}+\upsilon\frac{\partial u}{\partial y}=\nu\frac{\partial^2 u}{\partial y^2} \tag{8}$$

$$y=0: u=\upsilon=0; y=\infty: u=U_\infty \tag{9}$$

As velocity changes in the streamwise direction, velocity in the other directions will change as well. There is a small component of velocity at right angles to the surface which displacement the flow above it. The thickness of the boundary layer can be defined as the amount of this displacement.

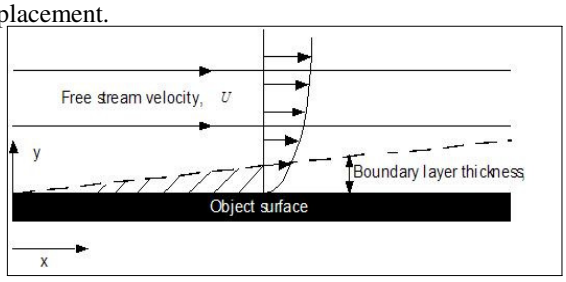

Figure 1. The boundary layer along a flat plate

III. NUMERICAL METHODS

A. Direct Method

A block tridiagonal matrix is obtained after we applied the Keller-box method on the boundary layer equation (8), which is having square matrices (blocks) in the lower, main, and upper diagonal, while all other blocks is a zero matrices. It is basically a tridiagonal matrix but has submatrices in places of scalars. A block tridiagonal matrix in this case study has the form as follow:

$$\begin{pmatrix} [A_1] & [C_1] & & & & \\ & [A_2] & [C_2] & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & [B_{J-1}] & [A_{J-1}] & [C_{J-1}] \\ & & & & & [B_J] & [A_J] \end{pmatrix} \begin{pmatrix} [\delta_1] \\ \vdots \\ \end{pmatrix} = \begin{pmatrix} [r_1] \\ \vdots \\ \end{pmatrix}$$

that is a linear system of equation

$$[A][\delta]=[r]. \tag{10}$$

1) LU Factorization

To solve equation (10) using LU factorization, we need to decompose A into a product of a lower triangular matrix, L and an upper triangular matrix, U as follows,

$$[A]=[L][U]. \tag{11}$$

where

$$[L]=\begin{pmatrix} [\alpha_1] & & & & & \\ & [\alpha_2] & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & [B_{J-1}] & [\alpha_{J-1}] \\ & & & & & [B_J] & [\alpha_J] \end{pmatrix}$$

and

$$[U]=\begin{pmatrix} [I] & [\Gamma_1] & & & & \\ & [I] & [\Gamma_2] & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & [I] & [\Gamma_{J-1}] \\ & & & & & [I] \end{pmatrix}$$

$[I]$ is the identity matrix of order 3 and $[\alpha_i],[\Gamma_i]$ are 3×3 matrices which the elements are determined by the following equations:

$$[\alpha_1] = [A_1] \tag{12}$$

$$[A_1][\Gamma_1] = [C_1] \tag{13}$$

$$[\alpha_j] = [A_j] - [B_j][\Gamma_{j-1}], j = 2,3,\ldots,J \tag{14}$$

$$[\alpha_j][\Gamma_j] = [C_j], j = 2,3,\ldots,J-1. \tag{15}$$

Equation (11) can be substituted into equation (10), and so we get

$$[L][U][\delta] = [r] \tag{16}$$

now define

$$[U][\delta] = [w], \tag{17}$$

then (16) becomes

$$[L][w] = [r], \tag{18}$$

where $[w]$ is a vector. The elements $[W]$ can be obtained from (18) as follows.

$$[\alpha_1][w_1] = [r_1], \tag{19}$$

$$[\alpha_j][w_j] = [r_j] - [B_j][w_{j-1}], 2 \le j \le J. \tag{20}$$

This step is referred to as a forward elimination. Once the elements of $[w]$ are found, (17) then gives the solution $[\delta]$ in the so-called backward substitution.When the elements of $[\delta]$ are found, Newton's method can be used to find the $(i+1)$ iteration. These calculations are repeated until some convergence criterion is satisfied.

2) Gauss Elimination

Another effective direct method to solve linear (10) is Gauss elimination . The objective of Gaussian elimination is to convert the general system of linear equations into a triangular system of equations [1]. The process of Gauss elimination has two parts. The first step is forward elimination reduces a given system to a triangular system. This is accomplished through the use of elementary row operations, which applies the characteristic of linear equations that any row can be replaced by a row that added to another row and multiplied by a constant. The second step uses back substitution to find the solution of the triangular system.

A. *Iterative Method*

3) Conjugate Gradient

The conjugate gradient method is an algorithm for the numerical solution of particular systems of linear equations. As it is an iterative method, so it can be applied to sparse systems that are too large to be handled by direct methods. The potential of conjugate gradient theoretically as a parallel algorithm has been experimented by [15] and [16].

4) Gauss Seidel Red-Black

Another popular iterative scheme to solve a system of linear equation is Gauss seidel. In this work, gauss seidel method is used to solve the problem sequentially. For parallel purpose, we applied other generation of gauss seidel namely Gauss seidel red-black which is more efficient when implemented into a parallel machine.

IV. Formulation Of Parallel Algorithm

A. *Parallel Direct Methods*

The data decomposition for direct LU factorization was design as Fig. 2.

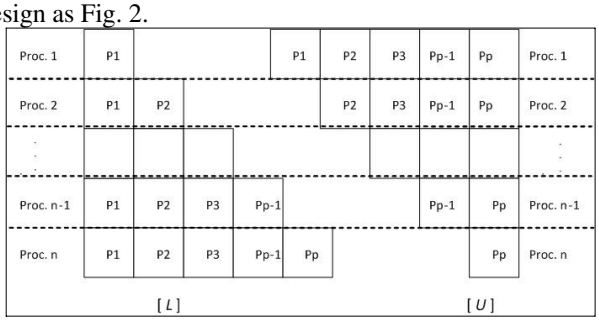

Figure 2. Data decomposition for calculating L and U matrices

The data is divided by rows. The P_1 to P_p are the processes involved in the parallel calculation where P_1 will be calculated by the processor 1 (Proc.1) and after the result of $[w_1]$ from the equation (19) is obtained, it will pass the result to the next processors (Proc.2,…,Proc.n) to calculate the elements of $[w_j]$. P_2 is solved by the Proc.2 and the result obtained is passed to the next processors (Proc.3,…,Proc.n). Once the last processor, Proc.n get the value of $[w_J]$, the result then will be used to continue the calculation for the (17) using the backward substitution.

As for Gauss elimination, the data decomposition is the same as LU factorization but only involved the U matrix which can be solve using the backward substitution. The implementation of LU and Gauss elimination on parallel computers is based on the pipeline configuration of the processors.

B. *Parallel Iterative Methods*

Gauss seidel red-black decompose domain Ω to two subdomain on red grid R, Ω^R and subdomain on black grid, B, Ω^B. Ω^R is an approximate solution on odd grid

and Ω^B is an approximation solution on even grid. The computation is first done on Ω^R and followed by computation on Ω^B. The decomposition of domain Ω to these two subdomain makes the computation on grid (i) is independent and easy to be implement on parallel computer system. The implementation of parallel conjugate gradient, can be develop directly without much modification on sequential CG. The non-overlapping subdomains of CG make it easy to distribute the data equally among all processors.

CG Algorithm {
compute $r_i^{(k)} \leftarrow b_i^{(k)} - Au_i^{(k)}$
if $r_i^{(k)} > \epsilon(p)$ then $d_i^{(k)} \leftarrow r_i^{(k)}$
communication between processors to send and receive $d_i^{(k)}$ from neighbor processor.
if $r_i^{(k)} > \epsilon(p)$ then $p_i^{(k)} \leftarrow Ad_i^{(k)}$
and set $\alpha_i^{(k)} \leftarrow (r_i^{(k)} \times r_i^{(k)})/(d_i^{(k)} \times p_i^{(k)})$,
$o_i^{(k)} \leftarrow u_i^{(k)} + \alpha_i^{(k)} \times d_i^{(k)}$, $R_i^{(k)} \leftarrow r_i^{(k)} - \alpha_i^{(k)} \times p_i^{(k)}$,
$\beta_i^{(k)} \leftarrow (R_i^{(k)} \times R_i^{(k)})/(r_i^{(k)} \times r_i^{(k)})$,
$D_{i \leftarrow R_i^{(k)}}^{(k)} + \beta_i^{(k)} \times d_i^{(k)}$
else
{
$o_i^{(k)} \leftarrow u_i^{(k)}$
}
if $r_i^{(k)} \leq \epsilon(p)$ then $d_i^{(k)} \leftarrow D_i^{(k)}$
and set $r_i^{(k)} \leftarrow R_i^{(k)}$, $u_i^{(k)} \leftarrow o_i^{(k)}$
}

Figure 3. Parallel algorithm for conjugate gradient method

V. COMPUTATIONAL COMPLEXITY

The computational complexity of direct methods and iterative methods are shown in Table 1 and 2 respectively. From the table, the minimum computational complexity is shown by Gauss elimination compared to LU factorization. As for the iterative methods, conjugate gradient shows the minimum computational complexity follows by GSRB method. In terms of rank, conjugate gradient will be the best method to solve the Keller-box matrix and followed by GSRB, Gauss elimination, and LU factorization.

TABLE I. COMPUTATIONAL COMPLEXITY OF PARALLEL DIRECT METHODS

Method	Multiplications	Additions
LU	$\frac{10m}{p}+13$	$\frac{10m}{p}+7$
Gauss Elimination	$\frac{10m}{p}+13$	$\frac{8m}{p}+7$

TABLE II. COMPUTATIONAL COMPLEXITY OF PARALLEL ITERATIVE METHODS

Method	Multiplications	Additions
GSRB	$8m+13$	$7m+7$
CG	$4m+13$	$9m+7$

VI. PARALLEL PERFORMANCE EVALUATION

Performance evaluation for parallel algorithm is measured in terms of speedup, efficiency, effectiveness, and temporal performance. The experimental results are based on 200000 size of matrix A.

A. Speedup and Efficiency

The speedup ($S(p)$), is a relative measurement of parallel algorithm performance over the sequential algorithm. If t_1 is an execution time of a sequential algorithm on one processor, and t_p is an execution time of parallel algorithm on p processors, then $S(p)$ formula is given by [13][14],

$$S(p) = \frac{t_1}{t_p} \tag{21}$$

Efficiency ($E(p)$) is a measurement that explained the speedup over the benefits of applying the parallel algorithm. $E(p)$ determined the ability of one algorithm using p number of processors. $E(p)$ is given by the formula [13][14],

$$E(p) = \frac{S(p)}{p} \tag{22}$$

and satisfy $0 \le E(p) \le 1$. If $S(p) = p$ then $E(p) = 1$, thus $E(p)$ is considered maximum. For the usual cases, $E(p)$ will decrease when number of processors is increased. $E(p)$ will be on the optimum level or will decrease slowly at a certain number of processors. Fig. 4 shows the speedup and efficiency for parallel direct methods. Based on the graphs, the speedup is increased when the number of processor is increase. The gradient of speedup is linear on $p < 12$, since of optimum load balancing and data distribution on all processors. The efficiency graph is incline when $p > 12$, as the processors need additional communication time to send and receive data, while idle time increase as the imbalance of the workload and also caused by the pipeline implementation on parallel algorithms of direct methods.

For the iterative methods on Fig. 4, conjugate gradient shows the great improvement on speedup compared to GSRB method. This proved that conjugate gradient is very suitable to be implement on parallel computers and to solve a large problems.

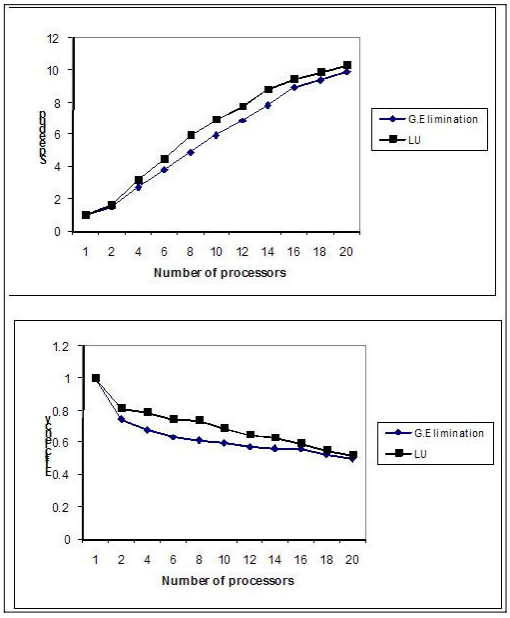

Figure 4. Speedup and efficiency for direct methods

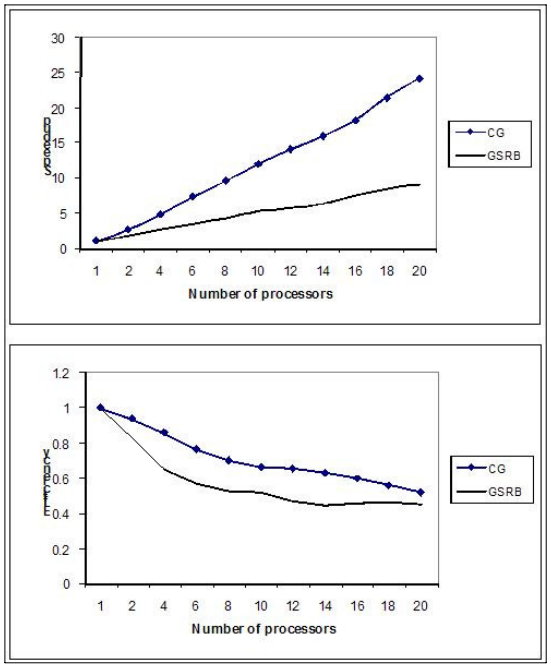

Figure 5. Speedup and efficiency for iterative methods

B. *Effectiveness and Temporal Performance*

Fig. 5 shows effectiveness and efficiency of parallel direct methods. The temporal performance of Gauss elimination is better than LU factorization when $p > 10$. For iterative methods on Fig. 6 shows the effectiveness and temporal performance of CG is much better than GSRB. This proved that CG has a very good performance in solving a large sparse problem, and can be an alternative solution to solve Keller-box matrix.

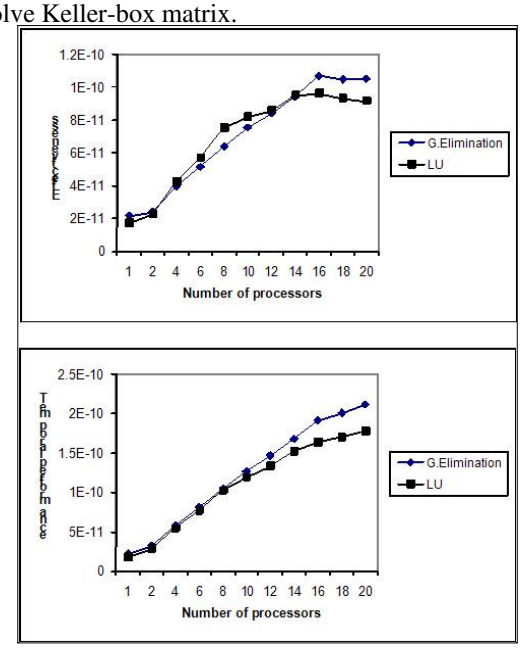

Figure 6. Effectiveness and temporal performance for direct methods

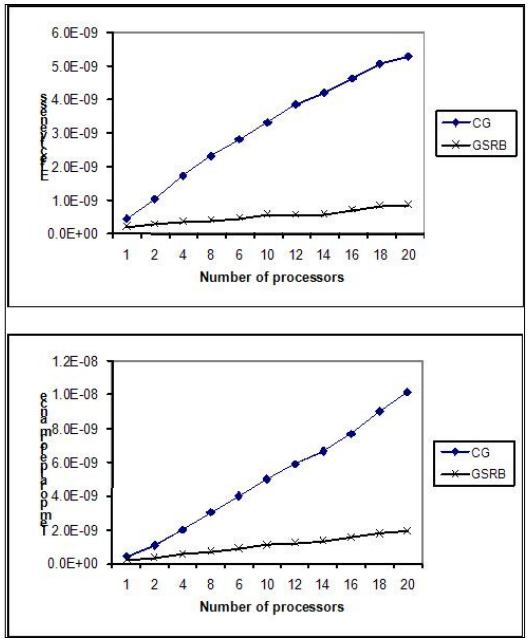

Figure 7. Effectiveness and temporal performance for iterative methods

VII. CONCLUSION

In this work, we have presented the experimental results illustrating the parallel implementation of iterative block and direct method using PVM programming environment. This paper contributes to the parallelization of iterative and direct method as an alternative to solve the large-sparse matrices for Keller-box method.

ACKNOWLEDGMENT

This work was supported in part by Research Management Center, UTM and Ministry of Science, Technology and Innovation Malaysia (MOSTI) through National Science Fellowship scholarship.

REFERENCES

[1] M. Rashid and C. Jon, Parallel iterative solution method for large sparse linear equation systems. 1em plus 0.5em minus 0.4em United Kingdom: University of Cambridge, 2005.

[2] J. M. George, "a new set of direct and iterative solvers for the tough2 family of codes,". 1em plus 0.5em minus 0.4em presented at the TOUGH workshop 95, Berkeley, 1995, p 293-298.

[3] W. E. Louis, "Iterative vs. a directive method for solving fourth order elliptic difference equations," 1em plus 0.5em minus 0.4em Proc. ACM national meeting, 1966, pp. 29-35.

[4] Y. Saad and H. A. van der Vorst, " Iterative solution of linear systems in the 20-th century," 1em plus 0.5em minus 0.4em J. Comp. Appl. Math., 2000, vol. 123, pp. 1-33.

[5] H. A. Frederick, "Fundamental of boundary layers (An NCFMF film)," 1em plus 0.5em minus 0.4em unpublished.

[6] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. Templates for the Solution of Linear Systems:Building Blocks for Iterative Methods. 1em plus 0.5em minus 0.4em Philadalphia: Society for Industrial and Applied Mathematics, 1994.

[7] H. Schlichting, Boundary layer theory 1em plus 0.5em minus 0.4em New York: McGraw-Hill, 1979, pp. 127-148.

[8] H. B. Keller, A new difference scheme for parabolic problems. In: Hubbard, B. ed. Numerical solutions of partial differential equations. 1em plus 0.5em minus 0.4em New York: Academic Press, 1971, 2: 327-350.

[9] H. B. Keller, and T. Cebeci, "A numerical methods for boundary layer flows, I: Two-dimensional laminar flows," 1em plus 0.5em minus 0.4em Proc. 2nd Int. Conf. on Numerical Methods in Fluid Dynamics, New York: Springer-verlag, 1971.

[10] H. B. Keller, and T. Cebeci, "Accurate numerical methods for boundary layer flows, II: Two-dimensional turbulent flows," 1em plus 0.5em minus 0.4em AIAA Journal. Vol. 10, 1972, pp. 1193-1199.

[11] T. Cebeci, and A. Smith, Analysis of turbulent boundary layers. 1em plus 0.5em minus 0.4em New York: Academic Press, 1974.

[12] B. Wilkinson, and M. Allen, Parallel programming: Techniques and applications using networked workstations and parallel computers. 1em plus 0.5em minus 0.4em New Jersey: Prentice hall, 1998.

[13] D. J. Evans, and M. S. Sahimi, "The alternating group explicit iterative method (AGE) to solve parabolic and hyperbolic partial differential equation," 1em plus 0.5em minus 0.4em Annual Review of Numerical Fluid Mechanics and Heat Transfer, 1989, Vol.2, pp. 283-389.

[14] H. El-Rewini, and T. G. Lewis, Distributed and parallel computing. 1em plus 0.5em minus 0.4em New York: Manning Publication, 1998.

[15] J. M. Ortega, "Ordering for parallel conjugate gradient preconditioner," 1em plus 0.5em minus 0.4em Siam J. on Scientific Computing, 1997, Vol.18(3), pp. 177-185.

[16] H. G. Roos, T. Lin β, "Gradient recovery for singularity pertubed boundary value problems 1: One dimensional convection diffusion," 1em plus 0.5em minus 0.4em Siam J. on Scientific Computing, 2001, Vol.66(2), pp. 163-178.